From Markdown to Blog Post: A ZZCOLLAB Conversion Workflow

Converting standalone documentation into reproducible blog posts with Quarto

Ronald 'Ryy' G. Thomas

2025-12-02

Table of contents

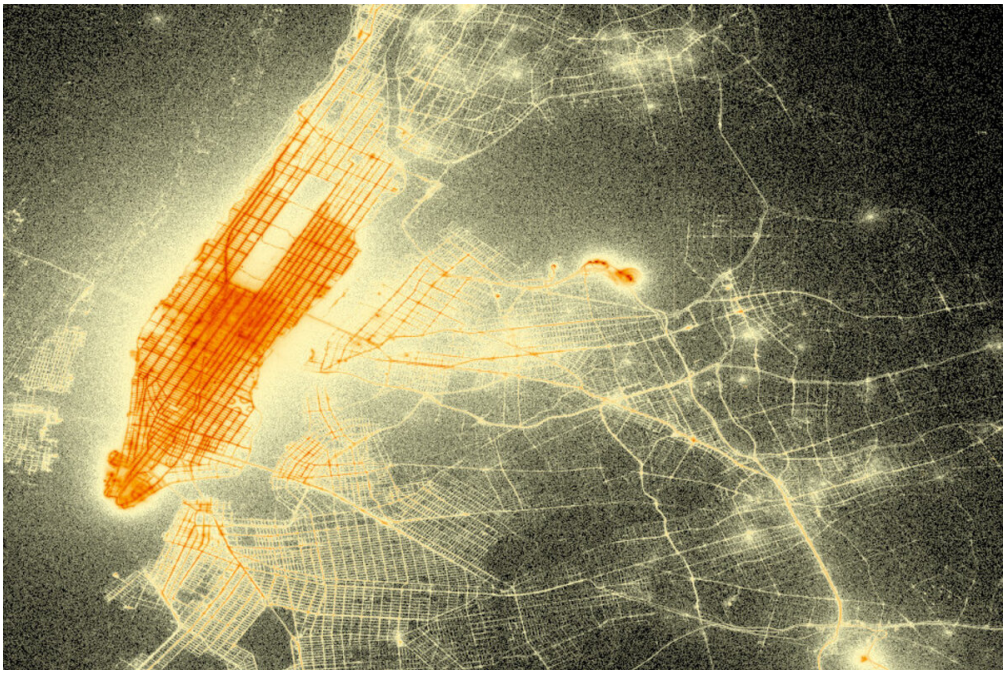

Figure 1: A workspace with documentation ready for conversion into a structured blog post

Transforming standalone documentation into reproducible, discoverable blog posts.

1 Introduction

Converting standalone markdown documentation into a professional blog post presents a common challenge: a well-written technical reference with working code examples and thorough coverage sits as a flat file on disk, invisible to anyone who might benefit from it.

The gap between a markdown file and a published blog post turns out to be narrower than expected, but only if a systematic workflow is followed. The ZZCOLLAB framework provides the scaffolding: symlinks bridge Quarto's expectations with the rrtools research compendium layout, and a reproducible Docker environment ensures that readers can rebuild the post from source.

This post documents that conversion workflow from start to finish, including the decisions required, the directory structure to create, and the verification steps that prevent broken links and missing images from reaching the audience.

More formally, this post documents the Document layer of the Workflow Construct described in post 52, specifically the operation that promotes a standalone Markdown document into a published blog post within the ZZCOLLAB compendium. The broader compendium-tier pattern is documented in post 29; this post is the conversion-specific complement, addressing the directory-structure, symlink, and YAML metadata work required to take an existing Markdown file and surface it as a published artefact.

1.1 Motivations

The following considerations motivated this workflow:

- Hundred-line markdown documents sitting in project directories remain invisible to anyone outside the project.
- Search engines cannot index a markdown file on a local disk; a published blog post reaches a broader audience.
- A repeatable process allows converting future documentation in minutes rather than hours.
- The ZZCOLLAB dual-symlink system already solves the Quarto-versus-rrtools path conflict; documenting its use for documentation conversion specifically fills a gap.
- A pre-publication checklist prevents publishing posts with broken image paths and missing YAML fields.

1.2 Objectives

1. Walk through the complete workflow for converting a standalone markdown file into a ZZCOLLAB-compatible Quarto blog post.
2. Explain the dual-symlink system that reconciles Quarto's directory expectations with the rrtools research compendium layout.
3. Provide concrete guidance on YAML frontmatter, media asset management, and content adaptation.
4. Deliver a pre-publication checklist covering content, metadata, media, structure, and testing.

Errors and better approaches are welcome; see the Feedback section at the end.

A structured workflow turns documentation into discoverable content.

2 Prerequisites and Setup

You will need the following tools installed before starting:

Required:

- ZZCOLLAB framework (installed and accessible via `zzcollab` command)

- Quarto (for rendering `.qmd` files to HTML)
- Git (for version control)
- Bash shell (macOS or Linux)

Optional but useful:

- GitHub CLI (`gh`) for automation
- Docker (for full reproducibility)
- A local Quarto preview server

Background assumed: Familiarity with markdown syntax, basic command-line operations, and the concept of symlinks. No prior experience with ZZCOLLAB is required; this post covers initialisation from scratch.

3 What is the ZZCOLLAB Blog Workflow?

The ZZCOLLAB blog workflow is a systematic process for converting standalone markdown documentation into a Quarto blog post that lives inside a reproducible research compendium. At its core, it solves a path conflict: Quarto expects blog posts at `posts/*/index.qmd`, while rrtools places authored content in `analysis/paper/`. The dual-symlink system bridges both conventions so that a single source file satisfies both tools.

Think of it as a translation layer. Your markdown content goes into the rrtools-standard location. Symlinks at the project root make Quarto believe the file is where it expects. Image paths like `media/images/hero.jpg` resolve correctly from both locations because symlinks exist in both directions.

The workflow itself takes roughly 40 minutes from start to finish: five minutes for assessment, three for initialisation, one for symlink setup, ten for content conversion, five for media assets, three for metadata, five for testing, and two for committing.

4 Getting Started

4.1 Step 1: Assess Your Source Document

Before touching any tools, spend five minutes evaluating the markdown file you plan to convert. Ask yourself:

1. How long is the document? A 200-line reference differs from a 600-line tutorial.
2. Does it contain code examples? If so, in which languages?
3. Does it need images? Hero images, diagrams, screenshots?
4. Is it a tutorial, reference guide, or how-to?
5. Who is the target audience?

```
Example assessment:
- Length: ~600 lines (substantial)
- Code: Yes (bash scripts, R snippets)
- Images: Yes (hero + concept diagram)
- Type: How-to guide with reference sections
- Audience: R developers using ZZCOLLAB
- Estimate: 10 minutes to convert content
```

4.2 Step 2: Create the ZZCOLLAB Project

```
cd ~/prj/qblog/posts
mkdir my_blog_post && cd my_blog_post
zzcollab -r ubuntu_standard_publishing
```

This creates the standard project structure: `analysis/`, `R/`, `tests/`, a Dockerfile, `renv.lock`, a Makefile, and `.zzcollab/manifest.json`.

4.3 Step 3: Set Up the Dual-Symlink System

```
./modules/setup_symlinks.sh
```

This creates symlinks at two levels:

At the project root (for Quarto discovery):

```
index.qmd  -> analysis/paper/index.qmd
figures/   -> analysis/figures/
media/     -> analysis/media/
data/      -> analysis/data/
```

Inside `analysis/paper/` (for intuitive editing):

```
figures/ -> ../figures/
media/   -> ../media/
data/    -> ../data/
```

The dual-symlink design means that when you write `![](media/images/hero.jpg)` in your blog post, the path resolves correctly whether Quarto renders from the project root or you preview from `analysis/paper/`.

Figure 2: An organised bookshelf with reference materials and a notebook

Proper directory structure prevents the most common blog post failures.

5 Converting Your Content

5.1 YAML Frontmatter

Replace the template YAML with metadata specific to your post. The fields that matter most for blog listing display are `title`, `categories`, `description`, and `image`:

```yaml
---
title: "Your Blog Post Title"
subtitle: "Clarifying subtitle"
author: "Your Name"
date: "2025-12-02"
categories: [quarto, zzcollab, reproducibility]
description: "2-3 sentence summary."
image: "media/images/hero.jpg"
document-type: "blog"
draft: false
execute:
  echo: true
  warning: false
  message: false
format:
  html:
    code-fold: false
    code-tools: false
---
```

Key points:

- Keep titles under 70 characters and include searchable keywords.
- Use 2-4 categories drawn from your blog's existing taxonomy.
- The `description` field appears in blog listings; lead with the value proposition.
- Set `draft: false` during development.

5.2 Adapting Markdown to Quarto

Most markdown syntax transfers directly. The main conversions to consider:

- **Blockquotes** become Quarto callouts:

```
::: {.callout-note}
## Summary
Your key takeaway goes here.
:::
```

- **Code blocks** should specify a language for syntax highlighting (`bash`, `r`, `python`).
- **Images** use the `.img-fluid` class for responsive display.

5.3 Adding Media Assets

Copy images into the media directory and document their sources:

```
cp ~/path/to/hero.jpg \
  analysis/media/images/hero.jpg

cat > analysis/media/images/README.md << 'EOF'
# Image Attribution

## hero.jpg
- Source: Unsplash
- Photographer: [Name]
- Licence: Unsplash Licence
- URL: [link]
EOF
```

Reference images in your post with relative paths:

```
![Alt text](media/images/hero.jpg){.img-fluid}

*Caption with attribution.*
```

6 Integration with the Parent Blog

The parent Quarto blog configuration (`_quarto.yml`) already renders posts matching the pattern `posts/*/index.qmd`. Because the root-level symlink maps `index.qmd` to `analysis/paper/index.qmd`, the blog discovers your post automatically.

No changes to the parent blog configuration are required. When you render the entire site with `quarto render`, the new post appears in the listing.

7 Version Control

Commit your work with a descriptive message that documents the content, media assets, and reproducibility infrastructure:

```
cd ~/prj/qblog/posts/my_blog_post
git add .
git status
git commit -m "Add blog post: Your Post Title
```

```
- analysis/paper/index.qmd: Main content
- analysis/media/images/: Hero and diagrams
- Dockerfile + renv.lock: Reproducible env"
```

Git handles symlinks correctly; they are stored as text files containing the target path.

8 Testing and Validation

8.1 Quick Validation

```
ls -la index.qmd
head -25 analysis/paper/index.qmd
grep "images/" analysis/paper/index.qmd
```

8.2 Local Render

```
cd ~/prj/qblog/posts/my_blog_post
quarto render analysis/paper/index.qmd
open index.html
```

Common issues and fixes:

Error	Cause	Fix
`index.qmd not found`	Broken symlink	`ls -la index.qmd`
Image not found	Wrong path	Use `media/images/file.jpg`
YAML parse error	Tabs in YAML	Use spaces only
No highlighting	Missing language	Specify ```` ```bash ````

8.3 Things to Watch Out For

1. **Symlinks break after cloning.** If someone clones your repository and the symlinks do not recreate correctly, they must run `setup_symlinks.sh` again or recreate them manually.
2. **Tabs in YAML cause silent failures.** Quarto YAML requires spaces for indentation. A single tab character produces a parse error that does not always point to the offending line.
3. **Absolute image paths break portability.** Always use relative paths (`media/images/hero.jpg`), never absolute paths. Absolute paths work on your machine and nowhere else.
4. **The `draft` field defaults to `false` when omitted.** If you forget to set `draft: false`, your unfinished post appears on the public site after the next render.
5. **Large images slow rendering and page load.** Keep images under 500 KB each. Use JPEG for photographs and PNG for diagrams with sharp edges.

Figure 3: A notebook with a completed checklist beside a cup of coffee

A pre-publication checklist prevents common deployment failures.

8.4 Lessons Learnt

8.4.1 Conceptual Understanding

- The dual-symlink system is the single most important architectural decision in the workflow; it reconciles two competing directory conventions without requiring changes to either tool.
- Blog posts are more discoverable than standalone markdown files; the effort of conversion pays off in audience reach.
- Structured YAML metadata enables Quarto's listing features (search, category filtering, date sorting) without custom JavaScript.
- The rrtools research compendium layout provides a natural separation between authored content, analysis code, and media assets.

8.4.2 Technical Skills

- Setting up symlinks with `ln -s` and verifying them with `ls -la` became second nature after the first conversion.

- Writing Quarto callouts (`::: {.callout-note} ... :::`) replaced blockquotes for emphasis and improved visual hierarchy.
- The `.img-fluid` class in Bootstrap ensures images scale correctly on mobile without any custom CSS.
- Using `quarto render` for local preview before committing caught broken paths that would otherwise reach the public site.

8.4.3 Gotchas and Pitfalls

- Forgetting to document image sources in `analysis/media/images/README.md` requires reconstructing attribution after the fact.
- A missing `.img-fluid` class causes hero images to overflow their container on mobile devices.
- Committing a post with `draft: false` by accident results in the post appearing live on the site before the metadata is corrected.
- Quarto's error messages for broken image paths are sometimes vague; using `grep` for all image references before rendering saves debugging time.

8.5 Limitations

- This workflow assumes a Unix-like environment (macOS or Linux) for symlink support. Windows users may need WSL or alternative approaches.
- The 40-minute estimate applies to straightforward documentation with minimal code and few images. Complex tutorials with interactive elements take longer.
- The dual-symlink system adds conceptual complexity that may confuse contributors unfamiliar with the convention.
- ZZCOLLAB must be installed and functional before starting. Troubleshooting ZZCOLLAB itself is outside the scope of this workflow.
- The workflow does not cover adding R analysis pipelines to posts that are purely narrative; extending a documentation post with data analysis requires additional steps.
- Local rendering requires Quarto installed on the author's machine. Docker-based rendering is possible but adds setup time.

8.6 Opportunities for Improvement

1. Create a shell script that automates the entire workflow: directory creation, ZZCOLLAB initialisation, symlink setup, and YAML template population from command-line arguments.
2. Build a validation script that checks all image paths, verifies symlink integrity, and confirms YAML field completeness before rendering.
3. Add a `make convert` target to the Makefile that accepts a source markdown path and performs the conversion automatically.
4. Develop a Quarto extension that validates `document-type` metadata and warns when required sections are missing.
5. Create a companion post covering the reverse workflow: extracting documentation from an existing blog post into a standalone reference.
6. Implement automated image optimisation (resizing, compression) as part of the media asset pipeline.

9 Wrapping Up

Converting standalone markdown documentation into a ZZCOLLAB blog post is a systematic, repeatable process that takes roughly 40 minutes once you understand the workflow. The dual-symlink system bridges the gap between Quarto's directory expectations and the rrtools research compendium layout, and proper YAML metadata ensures your content is discoverable through search and category filtering.

The key insight from developing this workflow is that the barrier to publishing is not the conversion itself but the lack of a structured process. Once the steps are documented and the checklist is in place, each subsequent conversion is faster and less error-prone.

For those with documentation sitting in project directories that could benefit a broader audience, this workflow merits consideration. The initial investment in understanding the symlink system pays dividends every time another document is converted.

Main takeaways:

- The dual-symlink system is the architectural foundation; understand it first.
- YAML metadata drives discoverability; invest time in titles, descriptions, and categories.
- A pre-publication checklist prevents the most common failures.
- The entire process takes about 40 minutes from assessment to committed post.

10 See Also

Related posts:

- ZZCOLLAB Blog Setup Guide: Comprehensive setup documentation
- Blog Post Template: The template this workflow produces

Key resources:

- ZZCOLLAB Framework: Project initialisation and management
- Quarto Blog Guide: Official Quarto blog documentation
- rrtools Research Compendium: The convention ZZCOLLAB follows
- Quarto YAML Reference: Complete YAML options

11 Reproducibility

This post is primarily narrative and does not involve a computational analysis pipeline. The ZZCOLLAB project structure provides the reproducibility infrastructure:

```
cd ~/prj/qblog/posts/06-markdowntoblog/
cd markdowntoblog
quarto render analysis/report/index.qmd
```

Project files:

```
markdowntoblog/
  analysis/report/index.qmd   # This post
  analysis/media/images/      # Hero, ambiance
  Dockerfile                  # Environment
  renv.lock                   # R packages
  Makefile                    # Build automation
```

12 Feedback

Feedback is welcome for:

- Errors or better approaches to any of the steps in this workflow
- Suggestions for topics to cover in future posts
- Discussion of documentation workflows, reproducibility, or Quarto publishing
- Questions about ZZCOLLAB or the symlink system
- General discussion of technical writing for data science